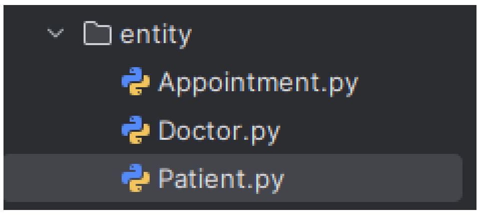
Coding Challenge: Hospital Management System

-Elakkiya

1Create SQL Schema from the following classes class, use the class attributes for table column names.

1. Create the following model/entity classes within package entity with variables declared private, constructors(default and parametrized,getters,setters and toString())



1. Define `Patient` class with the following confidential attributes:

a. patientId

b. firstName

c. lastName;

d. dateOfBirth

e. gender

f. contactNumber

g. address;

2. Define 'Doctor` class with the following confidential attributes:

a. doctorId

b. firstName

c. lastName

d. specialization

e. contactNumber;

3. Appointment Class:

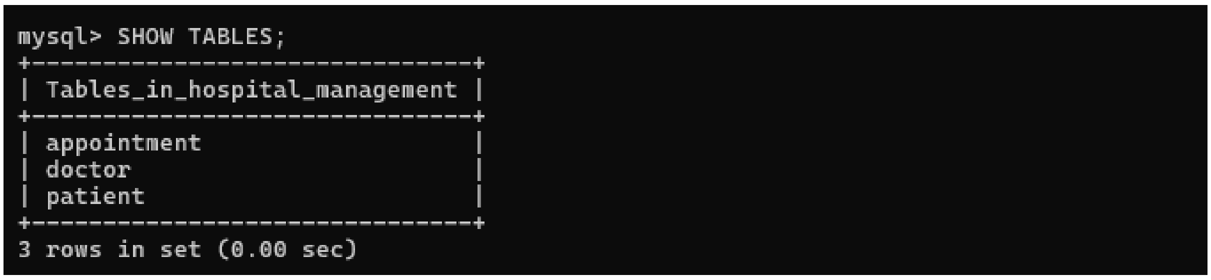
a. appointmentId

b. patientId

c. doctorId

d. appointmentDate

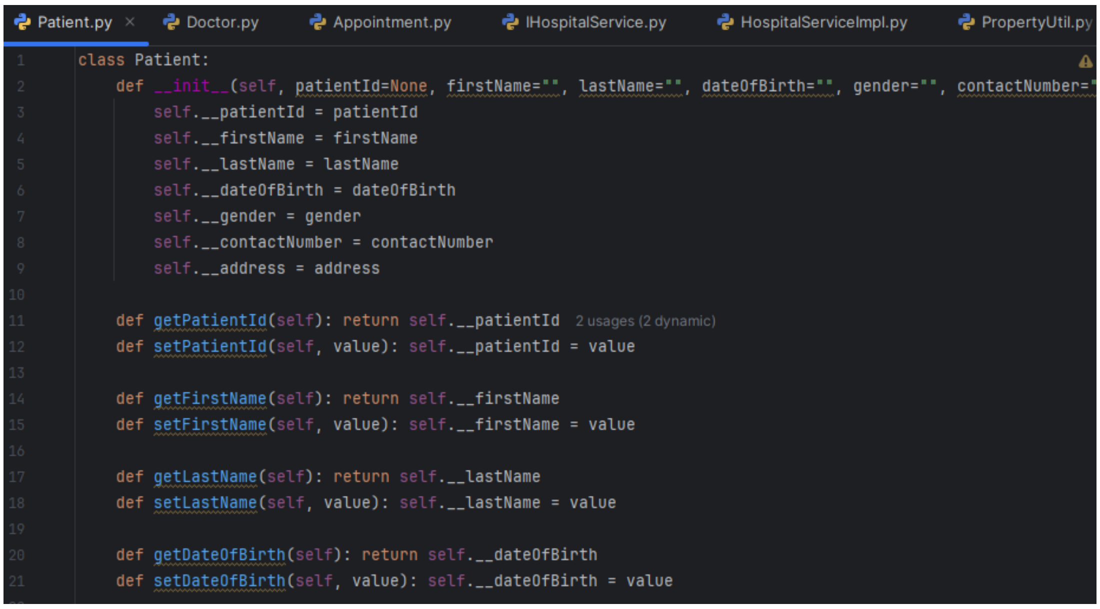
e. description



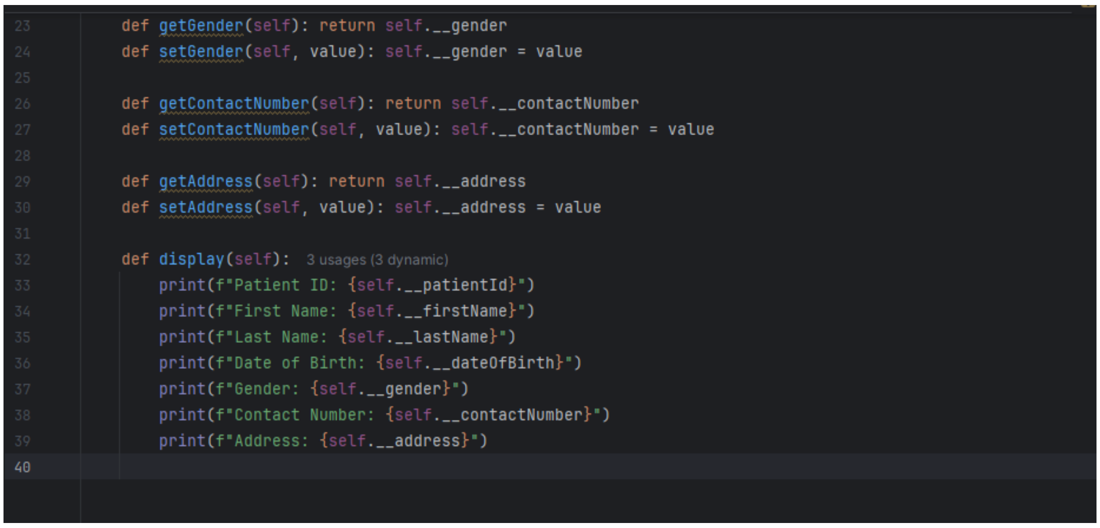
```
mysql> SHOW TABLES;
+------------------------------+
| Tables_in_hospital_management |
+------------------------------+
| appointment                   |
| doctor                        |
| patient                       |
+------------------------------+
3 rows in set (0.00 sec)
```

2. Implement the following for all model classes. Write default constructors and overload the constructor with parameters, getters and setters, method to print all the member variables and values.

Patient.py

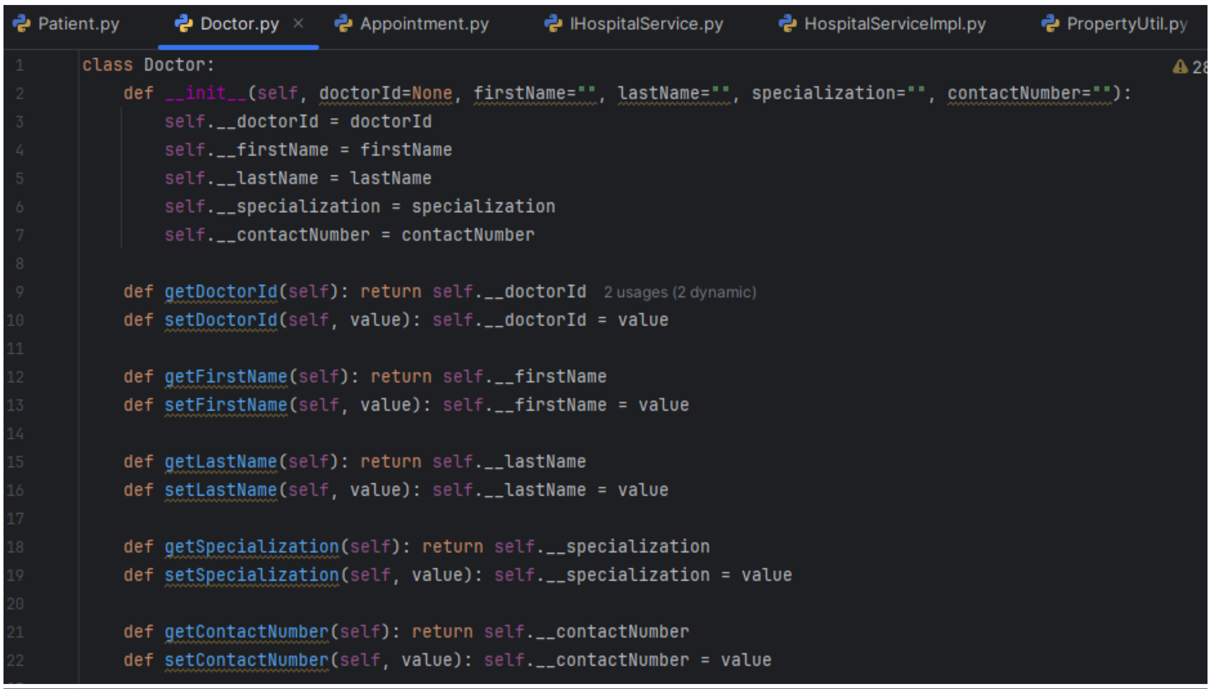


```python
class Patient:
    def __init__(self, patientId=None, firstName="", lastName="", dateOfBirth="", gender="", contactNumber="
        self.__patientId = patientId
        self.__firstName = firstName
        self.__lastName = lastName
        self.__dateOfBirth = dateOfBirth
        self.__gender = gender
        self.__contactNumber = contactNumber
        self.__address = address

    def getPatientId(self): return self.__patientId  2 usages (2 dynamic)
    def setPatientId(self, value): self.__patientId = value

    def getFirstName(self): return self.__firstName
    def setFirstName(self, value): self.__firstName = value

    def getLastName(self): return self.__lastName
    def setLastName(self, value): self.__lastName = value

    def getDateOfBirth(self): return self.__dateOfBirth
    def setDateOfBirth(self, value): self.__dateOfBirth = value
```

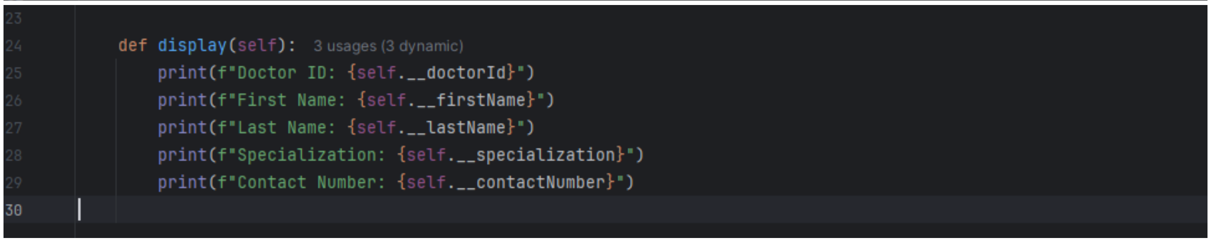


```python
    def getGender(self): return self.__gender
    def setGender(self, value): self.__gender = value

    def getContactNumber(self): return self.__contactNumber
    def setContactNumber(self, value): self.__contactNumber = value

    def getAddress(self): return self.__address
    def setAddress(self, value): self.__address = value

    def display(self):  3 usages (3 dynamic)
        print(f"Patient ID: {self.__patientId}")
        print(f"First Name: {self.__firstName}")
        print(f"Last Name: {self.__lastName}")
        print(f"Date of Birth: {self.__dateOfBirth}")
        print(f"Gender: {self.__gender}")
        print(f"Contact Number: {self.__contactNumber}")
        print(f"Address: {self.__address}")
```

Doctor.py

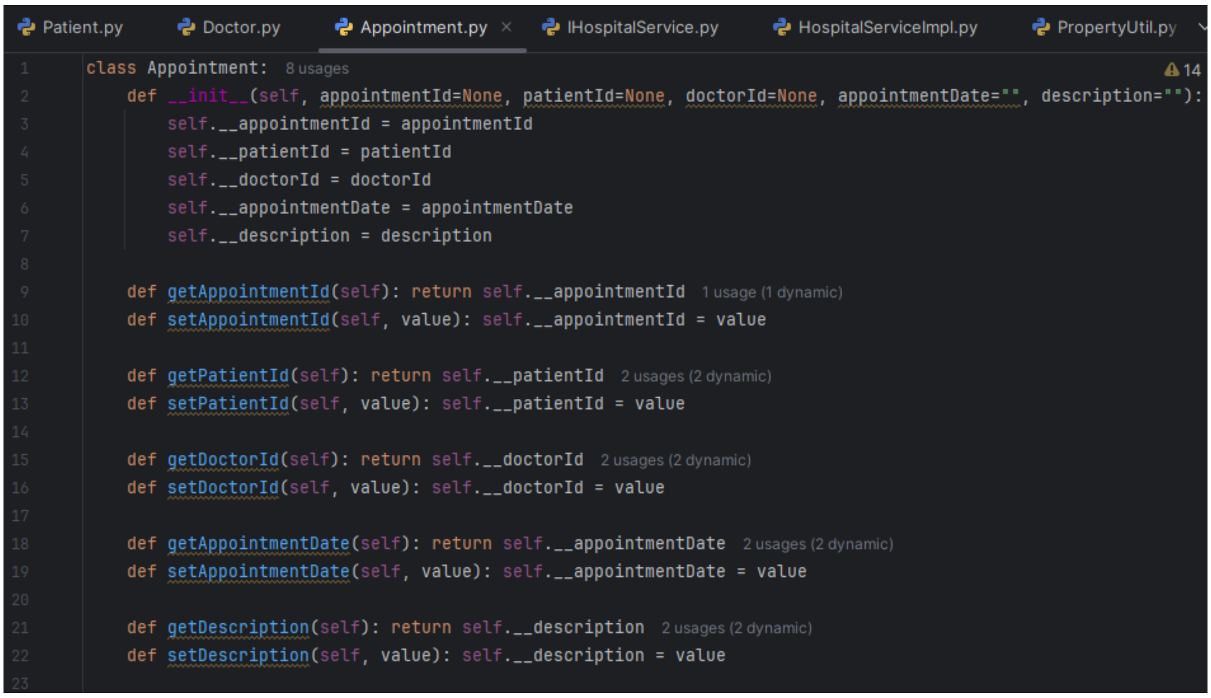


```python
class Doctor:
    def __init__(self, doctorId=None, firstName="", lastName="", specialization="", contactNumber=""):
        self.__doctorId = doctorId
        self.__firstName = firstName
        self.__lastName = lastName
        self.__specialization = specialization
        self.__contactNumber = contactNumber

    def getDoctorId(self): return self.__doctorId
    def setDoctorId(self, value): self.__doctorId = value

    def getFirstName(self): return self.__firstName
    def setFirstName(self, value): self.__firstName = value

    def getLastName(self): return self.__lastName
    def setLastName(self, value): self.__lastName = value

    def getSpecialization(self): return self.__specialization
    def setSpecialization(self, value): self.__specialization = value

    def getContactNumber(self): return self.__contactNumber
    def setContactNumber(self, value): self.__contactNumber = value
```

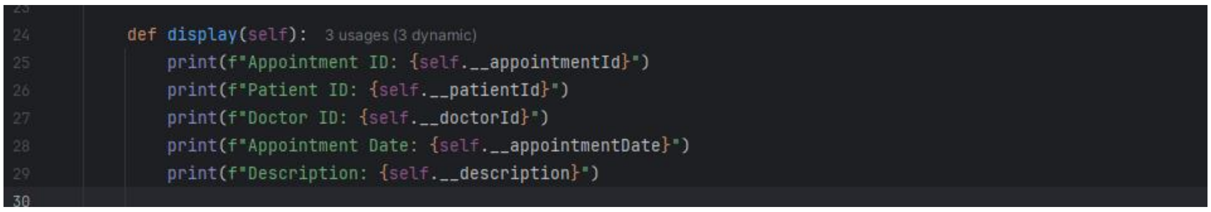


```python

    def display(self):
        print(f"Doctor ID: {self.__doctorId}")
        print(f"First Name: {self.__firstName}")
        print(f"Last Name: {self.__lastName}")
        print(f"Specialization: {self.__specialization}")
        print(f"Contact Number: {self.__contactNumber}")

```

Appointment.py



```python
class Appointment:
    def __init__(self, appointmentId=None, patientId=None, doctorId=None, appointmentDate="", description=""):
        self.__appointmentId = appointmentId
        self.__patientId = patientId
        self.__doctorId = doctorId
        self.__appointmentDate = appointmentDate
        self.__description = description

    def getAppointmentId(self): return self.__appointmentId
    def setAppointmentId(self, value): self.__appointmentId = value

    def getPatientId(self): return self.__patientId
    def setPatientId(self, value): self.__patientId = value

    def getDoctorId(self): return self.__doctorId
    def setDoctorId(self, value): self.__doctorId = value

    def getAppointmentDate(self): return self.__appointmentDate
    def setAppointmentDate(self, value): self.__appointmentDate = value

    def getDescription(self): return self.__description
    def setDescription(self, value): self.__description = value

```


```
    def display(self):  3 usages (3 dynamic)
        print(f"Appointment ID: {self.__appointmentId}")
        print(f"Patient ID: {self.__patientId}")
        print(f"Doctor ID: {self.__doctorId}")
        print(f"Appointment Date: {self.__appointmentDate}")
        print(f"Description: {self.__description}")
```

3. Define IHospitalService interface/abstract class with following methods to interact with database Keep the interfaces and implementation classes in package dao

a. getAppointmentById()

i. Parameters: appointmentId

ii. ReturnType: Appointment object

b. getAppointmentsForPatient()

i. Parameters: patientId

ii. ReturnType: List of Appointment objects

c. getAppointmentsForDoctor()

i. Parameters: doctorId

ii. ReturnType: List of Appointment objects

d. scheduleAppointment()

i. Parameters: Appointment Object

ii. ReturnType: Boolean

e. updateAppointment()

i. Parameters: Appointment Object

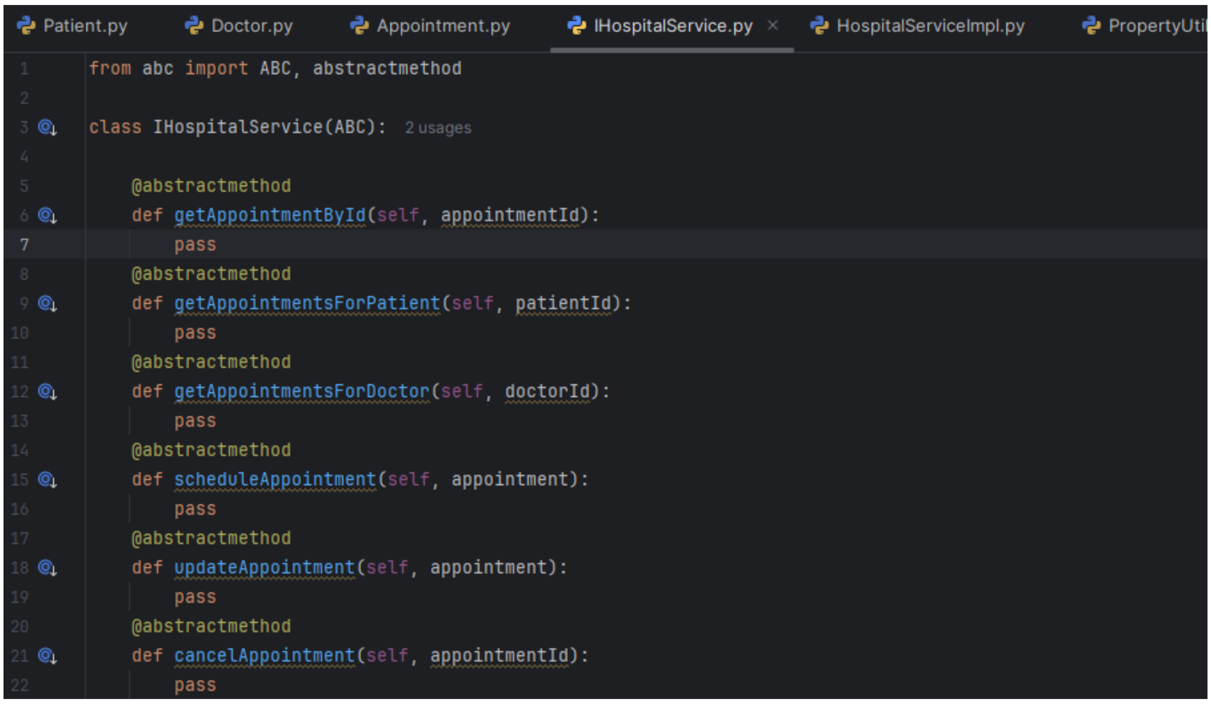
ii. ReturnType: Boolean

f. ancelAppointment()

i. Parameters: AppointmentId

ii. ReturnType: Boolean

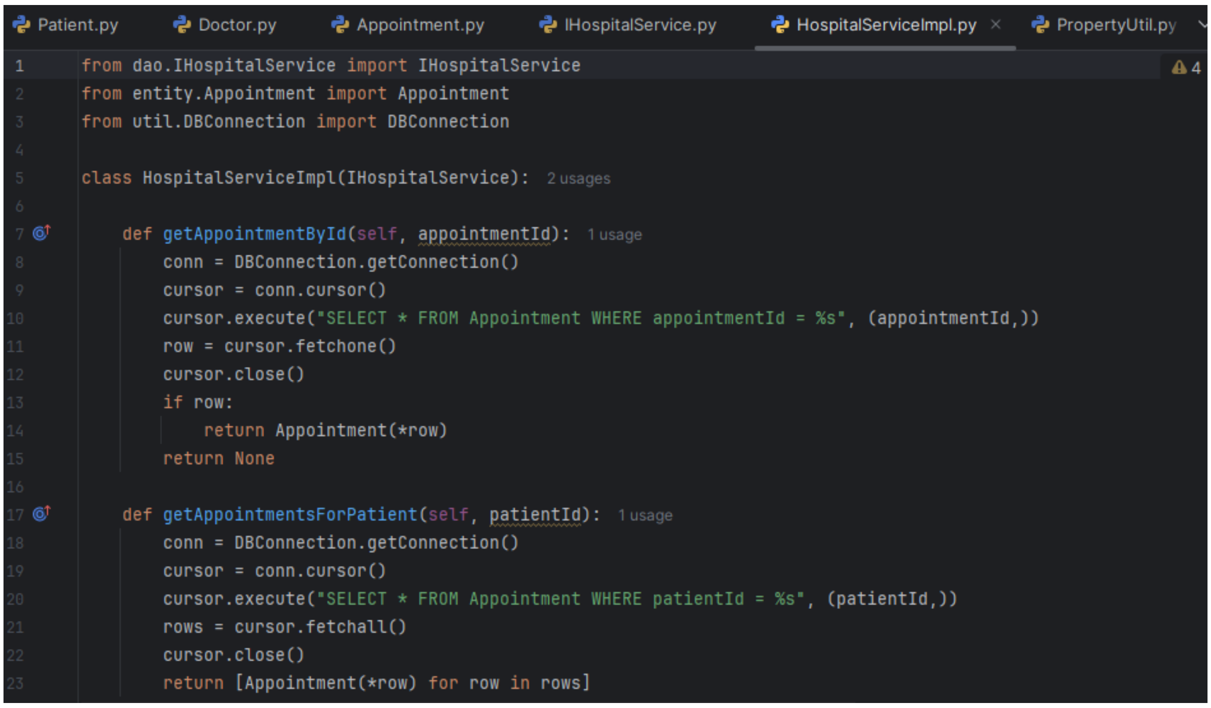
IHospitalService.py



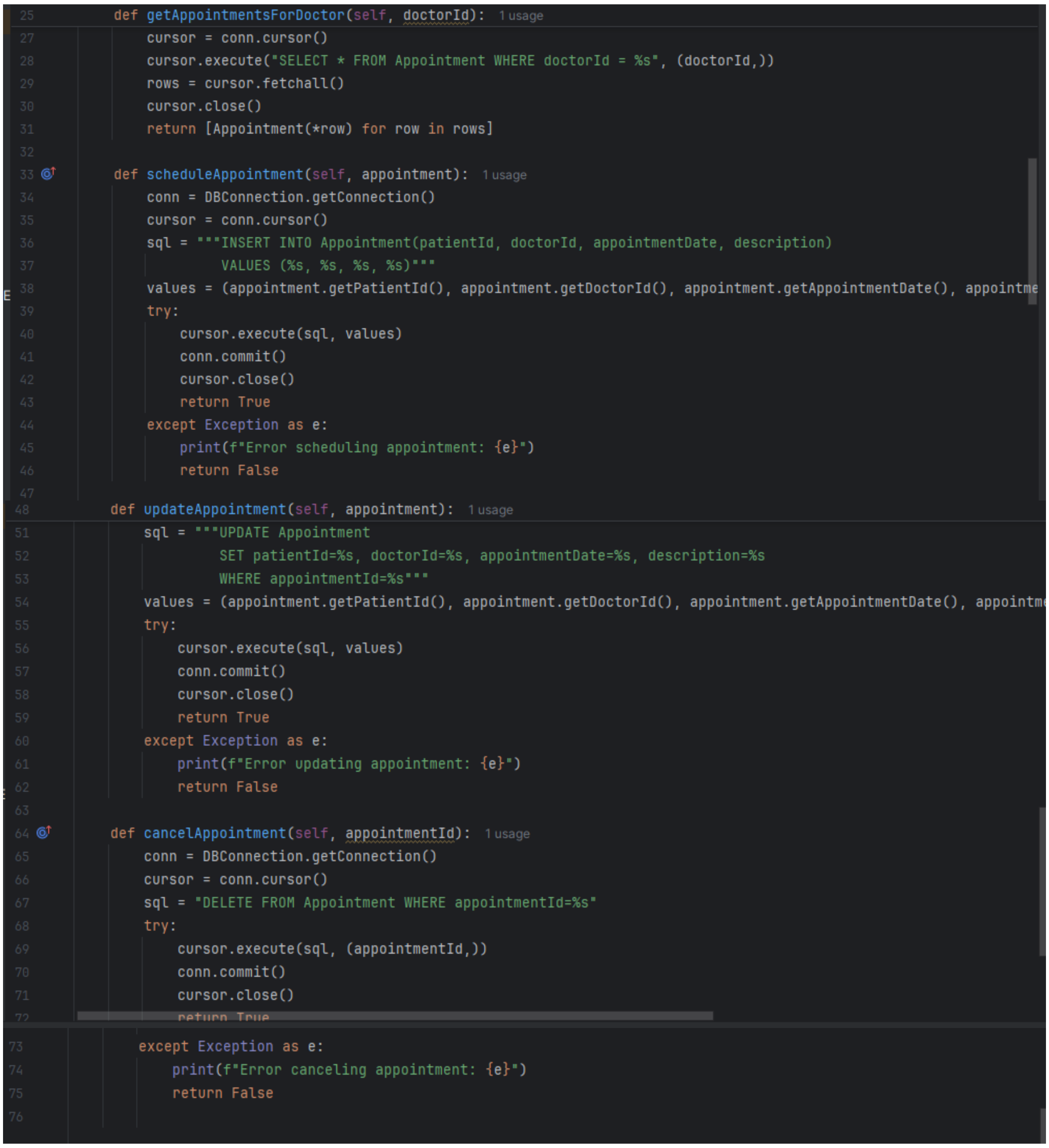
```python
from abc import ABC, abstractmethod

class IHospitalService(ABC):  2 usages

    @abstractmethod
    def getAppointmentById(self, appointmentId):
        pass
    @abstractmethod
    def getAppointmentsForPatient(self, patientId):
        pass
    @abstractmethod
    def getAppointmentsForDoctor(self, doctorId):
        pass
    @abstractmethod
    def scheduleAppointment(self, appointment):
        pass
    @abstractmethod
    def updateAppointment(self, appointment):
        pass
    @abstractmethod
    def cancelAppointment(self, appointmentId):
        pass
```

6. Define HospitalServiceImpl class and implement all the methods IHospitalServiceImpl.

HospitalServiceImpl.py



```python
from dao.IHospitalService import IHospitalService
from entity.Appointment import Appointment
from util.DBConnection import DBConnection

class HospitalServiceImpl(IHospitalService):  2 usages

    def getAppointmentById(self, appointmentId):  1 usage
        conn = DBConnection.getConnection()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Appointment WHERE appointmentId = %s", (appointmentId,))
        row = cursor.fetchone()
        cursor.close()
        if row:
            return Appointment(*row)
        return None

    def getAppointmentsForPatient(self, patientId):  1 usage
        conn = DBConnection.getConnection()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Appointment WHERE patientId = %s", (patientId,))
        rows = cursor.fetchall()
        cursor.close()
        return [Appointment(*row) for row in rows]
```

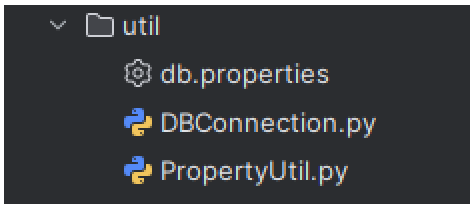


```python
    def getAppointmentsForDoctor(self, doctorId):  1 usage
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Appointment WHERE doctorId = %s", (doctorId,))
        rows = cursor.fetchall()
        cursor.close()
        return [Appointment(*row) for row in rows]

    def scheduleAppointment(self, appointment):  1 usage
        conn = DBConnection.getConnection()
        cursor = conn.cursor()
        sql = """INSERT INTO Appointment(patientId, doctorId, appointmentDate, description)
                 VALUES (%s, %s, %s, %s)"""
        values = (appointment.getPatientId(), appointment.getDoctorId(), appointment.getAppointmentDate(), appointme
        try:
            cursor.execute(sql, values)
            conn.commit()
            cursor.close()
            return True
        except Exception as e:
            print(f"Error scheduling appointment: {e}")
            return False

    def updateAppointment(self, appointment):  1 usage
        sql = """UPDATE Appointment
                 SET patientId=%s, doctorId=%s, appointmentDate=%s, description=%s
                 WHERE appointmentId=%s"""
        values = (appointment.getPatientId(), appointment.getDoctorId(), appointment.getAppointmentDate(), appointme
        try:
            cursor.execute(sql, values)
            conn.commit()
            cursor.close()
            return True
        except Exception as e:
            print(f"Error updating appointment: {e}")
            return False

    def cancelAppointment(self, appointmentId):  1 usage
        conn = DBConnection.getConnection()
        cursor = conn.cursor()
        sql = "DELETE FROM Appointment WHERE appointmentId=%s"
        try:
            cursor.execute(sql, (appointmentId,))
            conn.commit()
            cursor.close()
            return True
        except Exception as e:
            print(f"Error canceling appointment: {e}")
            return False
```

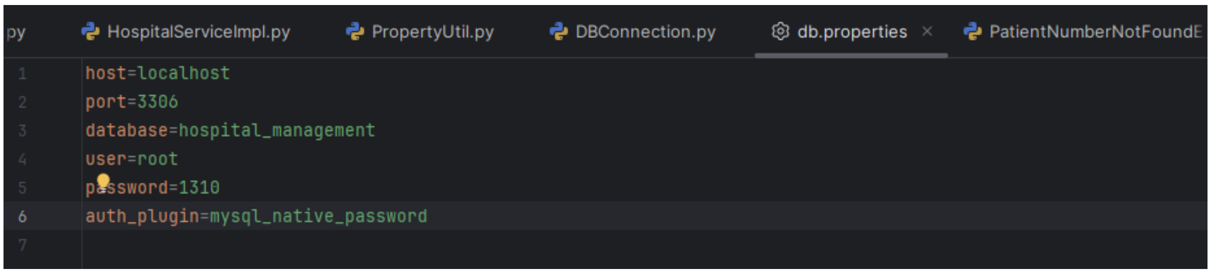
7. Create a utility class DBConnection in a package util with a static variable connection of Type Connection and a static method getConnection() which returns connection.

Connection properties supplied in the connection string should be read from a property file.

Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

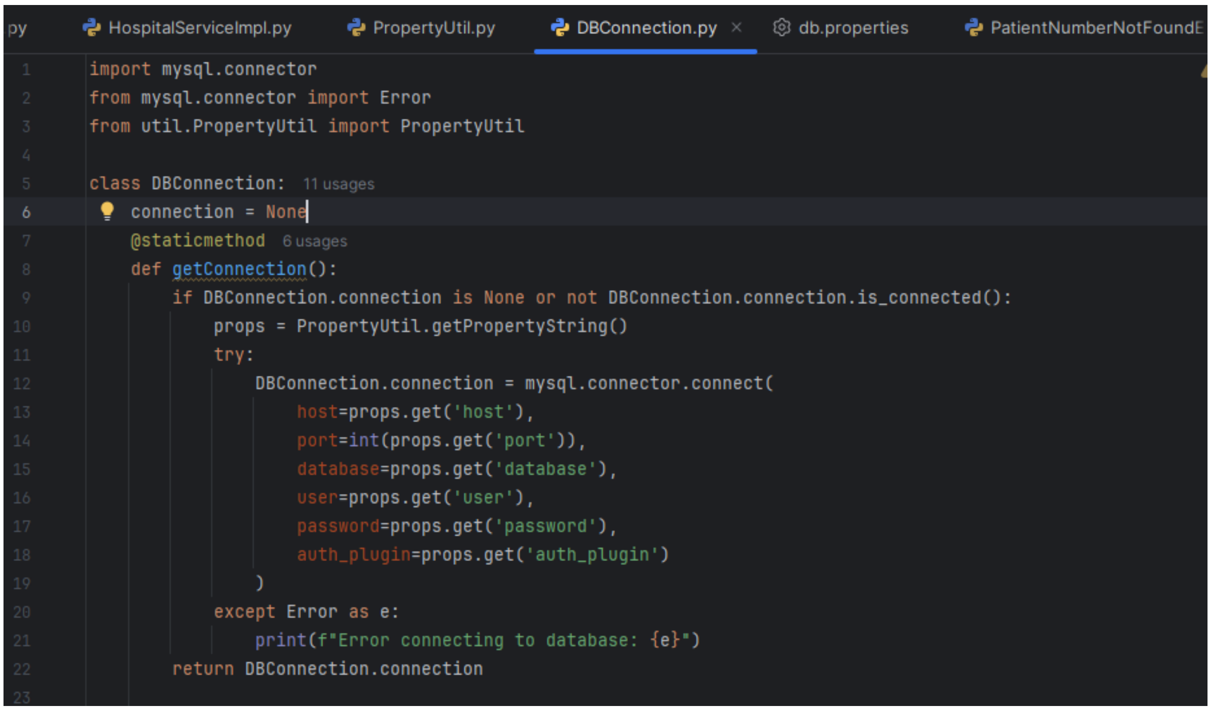


db.properties.py



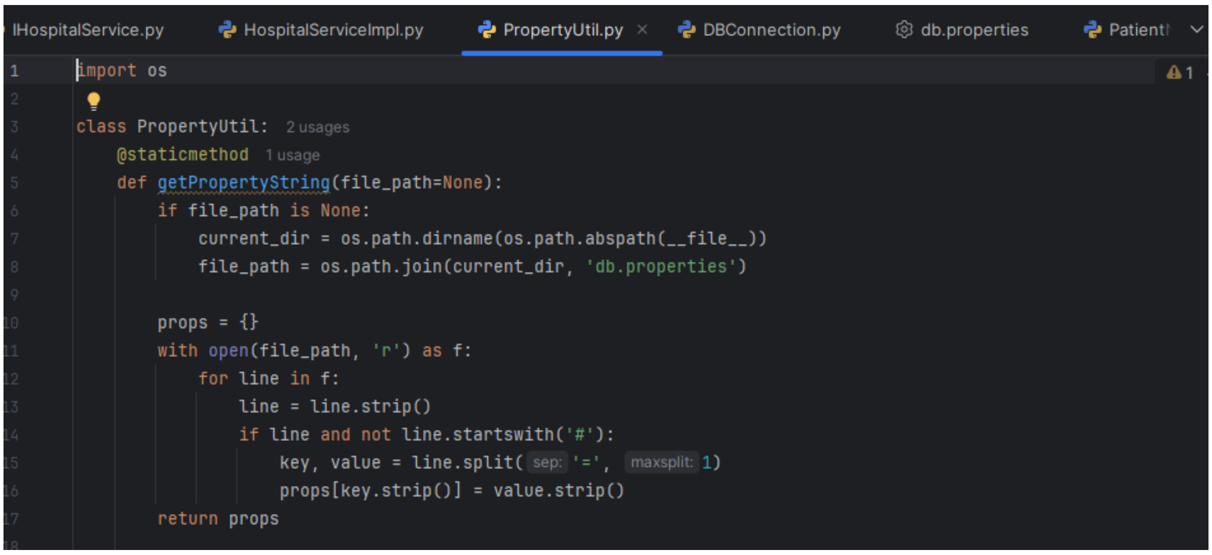
```
host=localhost
port=3306
database=hospital_management
user=root
password=1310
auth_plugin=mysql_native_password
```

DbConnection.py



```python
import mysql.connector
from mysql.connector import Error
from util.PropertyUtil import PropertyUtil

class DBConnection:  11 usages
    connection = None
    @staticmethod  6 usages
    def getConnection():
        if DBConnection.connection is None or not DBConnection.connection.is_connected():
            props = PropertyUtil.getPropertyString()
            try:
                DBConnection.connection = mysql.connector.connect(
                    host=props.get('host'),
                    port=int(props.get('port')),
                    database=props.get('database'),
                    user=props.get('user'),
                    password=props.get('password'),
                    auth_plugin=props.get('auth_plugin')
                )
            except Error as e:
                print(f"Error connecting to database: {e}")
        return DBConnection.connection
```

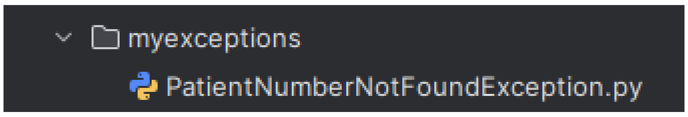
PropertyUtil.py

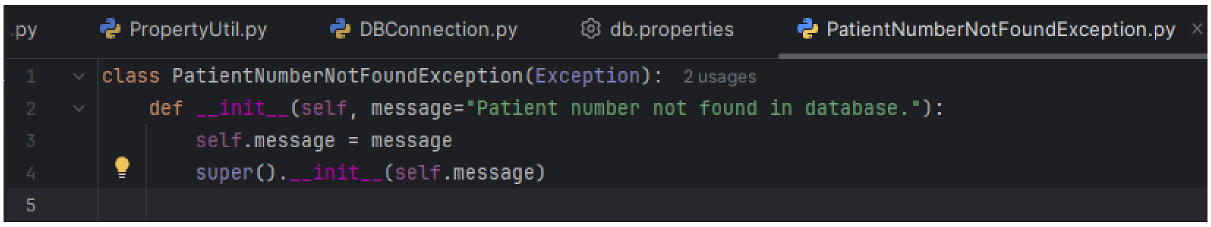


```
import os

class PropertyUtil:  2 usages
    @staticmethod  1 usage
    def getPropertyString(file_path=None):
        if file_path is None:
            current_dir = os.path.dirname(os.path.abspath(__file__))
            file_path = os.path.join(current_dir, 'db.properties')

        props = {}
        with open(file_path, 'r') as f:
            for line in f:
                line = line.strip()
                if line and not line.startswith('#'):
                    key, value = line.split( sep: '=',  maxsplit: 1)
                    props[key.strip()] = value.strip()
        return props
```

8. Create the exceptions in package myexceptions Define the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

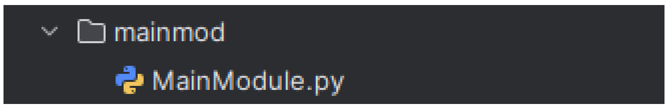
1. PatientNumberNotFoundException :throw this exception when user enters an invalid patient number which doesn’t exist in db

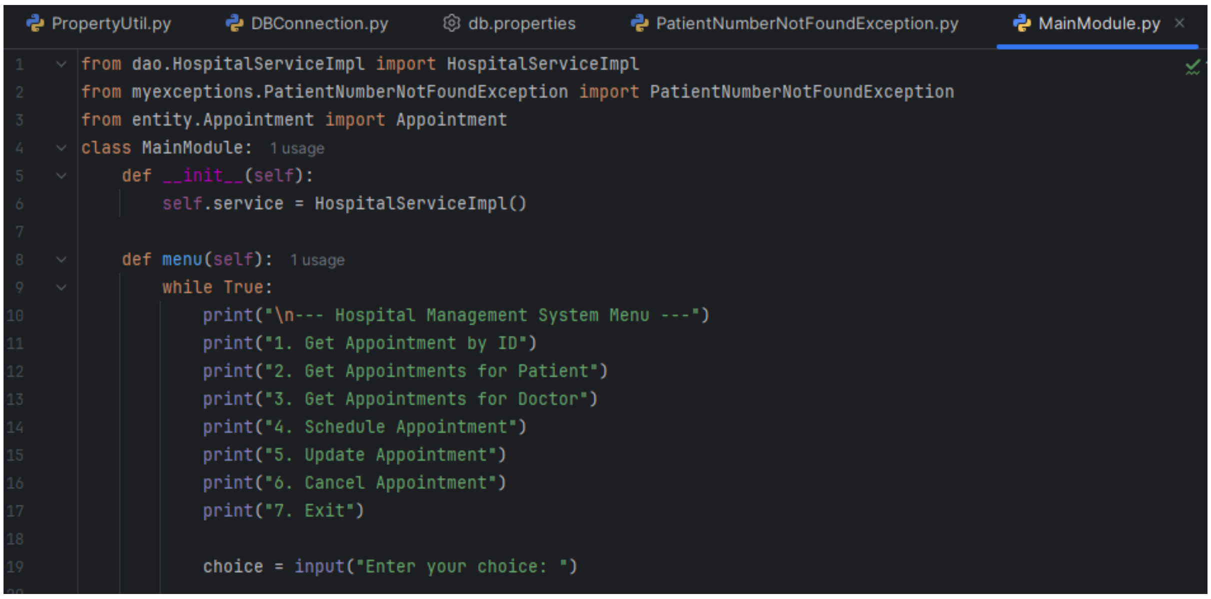




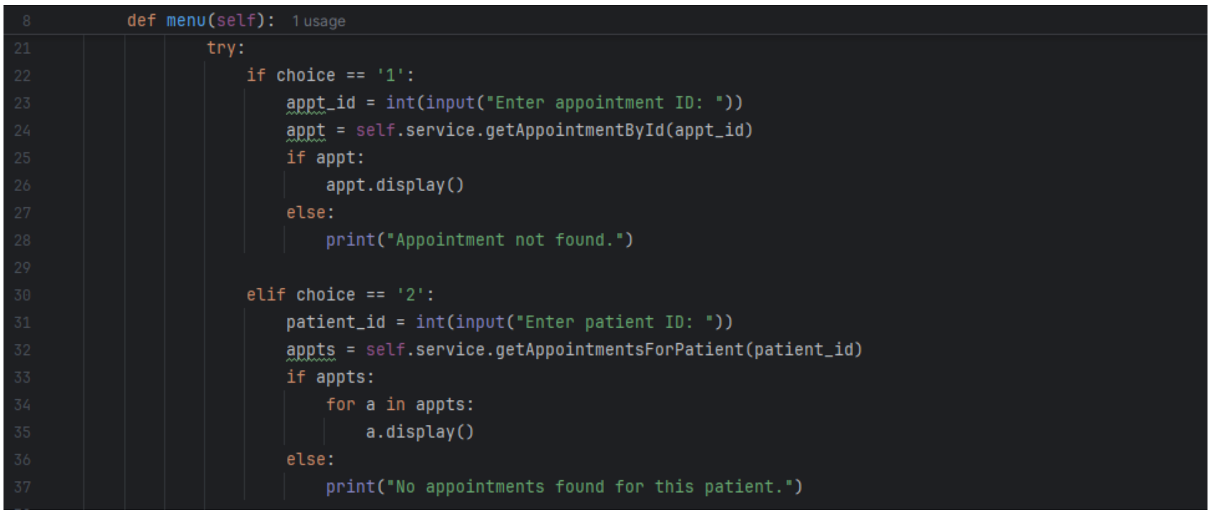
```
class PatientNumberNotFoundException(Exception):  2 usages
    def __init__(self, message="Patient number not found in database."):
        self.message = message
        super().__init__(self.message)
```

9. Create class named MainModule with main method in package mainmod. Trigger all the methods in service implementation class.

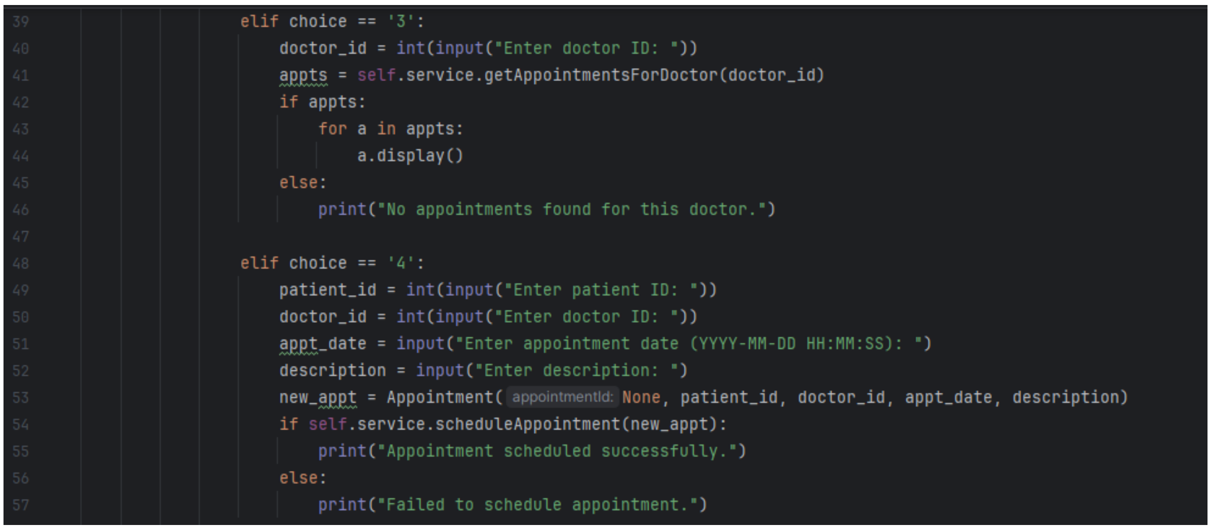




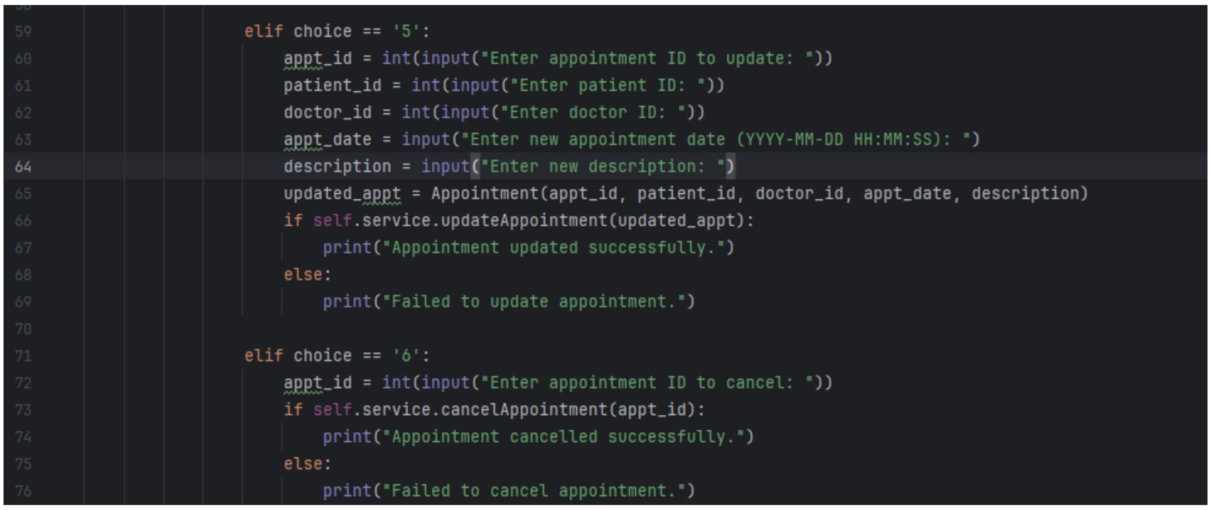
```python
from dao.HospitalServiceImpl import HospitalServiceImpl
from myexceptions.PatientNumberNotFoundException import PatientNumberNotFoundException
from entity.Appointment import Appointment
class MainModule:  1 usage
    def __init__(self):
        self.service = HospitalServiceImpl()

    def menu(self):  1 usage
        while True:
            print("\n--- Hospital Management System Menu ---")
            print("1. Get Appointment by ID")
            print("2. Get Appointments for Patient")
            print("3. Get Appointments for Doctor")
            print("4. Schedule Appointment")
            print("5. Update Appointment")
            print("6. Cancel Appointment")
            print("7. Exit")

            choice = input("Enter your choice: ")
```



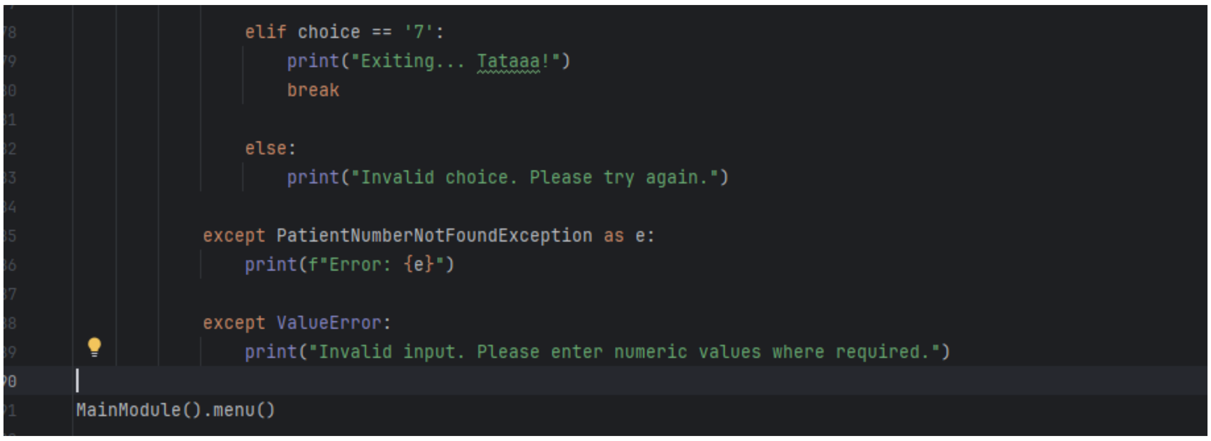
```python
    def menu(self):  1 usage
            try:
                if choice == '1':
                    appt_id = int(input("Enter appointment ID: "))
                    appt = self.service.getAppointmentById(appt_id)
                    if appt:
                        appt.display()
                    else:
                        print("Appointment not found.")

                elif choice == '2':
                    patient_id = int(input("Enter patient ID: "))
                    appts = self.service.getAppointmentsForPatient(patient_id)
                    if appts:
                        for a in appts:
                            a.display()
                    else:
                        print("No appointments found for this patient.")
```



```python
                elif choice == '3':
                    doctor_id = int(input("Enter doctor ID: "))
                    appts = self.service.getAppointmentsForDoctor(doctor_id)
                    if appts:
                        for a in appts:
                            a.display()
                    else:
                        print("No appointments found for this doctor.")

                elif choice == '4':
                    patient_id = int(input("Enter patient ID: "))
                    doctor_id = int(input("Enter doctor ID: "))
                    appt_date = input("Enter appointment date (YYYY-MM-DD HH:MM:SS): ")
                    description = input("Enter description: ")
                    new_appt = Appointment( appointmentId: None, patient_id, doctor_id, appt_date, description)
                    if self.service.scheduleAppointment(new_appt):
                        print("Appointment scheduled successfully.")
                    else:
                        print("Failed to schedule appointment.")
```

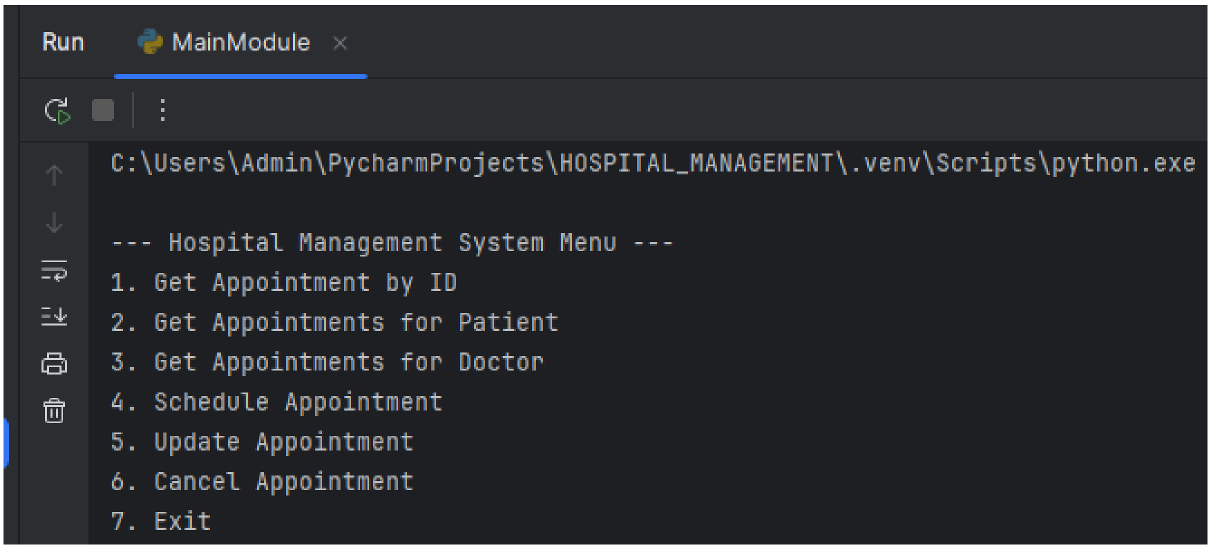


```
            elif choice == '5':
                appt_id = int(input("Enter appointment ID to update: "))
                patient_id = int(input("Enter patient ID: "))
                doctor_id = int(input("Enter doctor ID: "))
                appt_date = input("Enter new appointment date (YYYY-MM-DD HH:MM:SS): ")
                description = input("Enter new description: ")
                updated_appt = Appointment(appt_id, patient_id, doctor_id, appt_date, description)
                if self.service.updateAppointment(updated_appt):
                    print("Appointment updated successfully.")
                else:
                    print("Failed to update appointment.")

            elif choice == '6':
                appt_id = int(input("Enter appointment ID to cancel: "))
                if self.service.cancelAppointment(appt_id):
                    print("Appointment cancelled successfully.")
                else:
                    print("Failed to cancel appointment.")
```



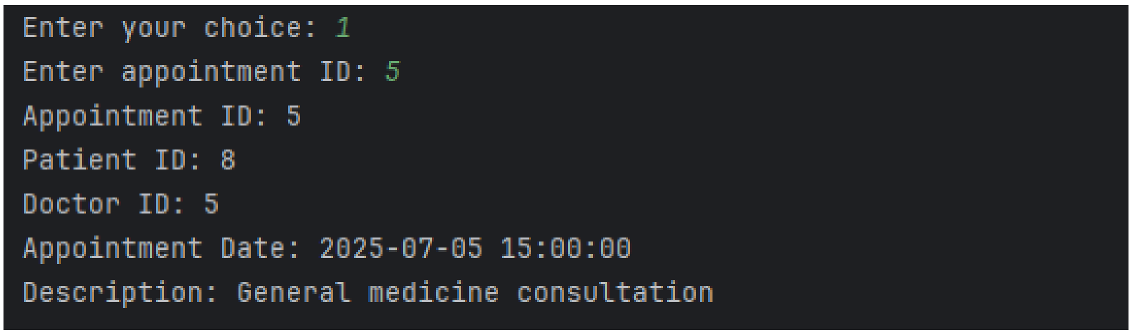
```
            elif choice == '7':
                print("Exiting... Tataaa!")
                break

            else:
                print("Invalid choice. Please try again.")

        except PatientNumberNotFoundException as e:
            print(f"Error: {e}")

        except ValueError:
            print("Invalid input. Please enter numeric values where required.")

MainModule().menu()
```

Output:



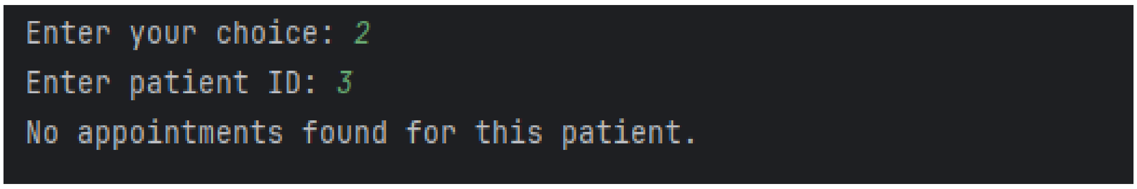
```
C:\Users\Admin\PycharmProjects\HOSPITAL_MANAGEMENT\.venv\Scripts\python.exe

--- Hospital Management System Menu ---
1. Get Appointment by ID
2. Get Appointments for Patient
3. Get Appointments for Doctor
4. Schedule Appointment
5. Update Appointment
6. Cancel Appointment
7. Exit
```

Get appointment by ID:



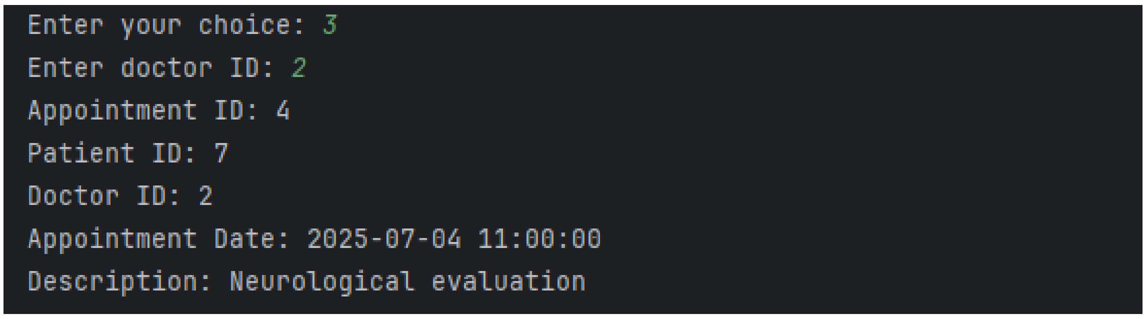
```
Enter your choice: 1
Enter appointment ID: 5
Appointment ID: 5
Patient ID: 8
Doctor ID: 5
Appointment Date: 2025-07-05 15:00:00
Description: General medicine consultation
```

Get appointments for patient:



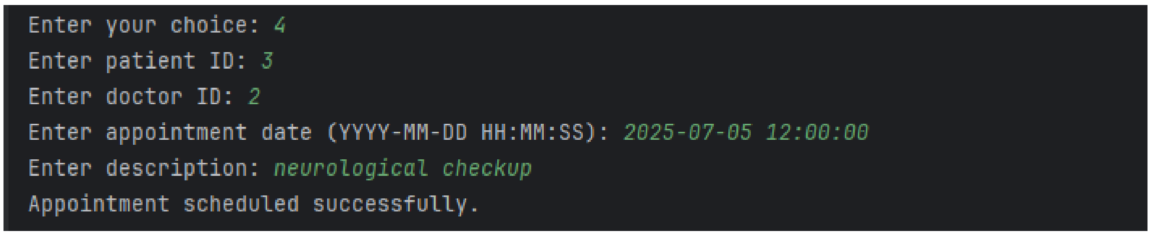
```
Enter your choice: 2
Enter patient ID: 3
No appointments found for this patient.
```

Get appointments for doctor:



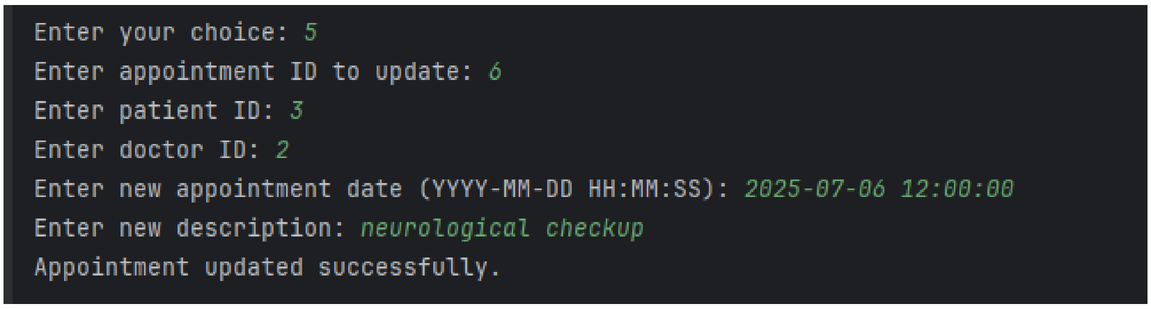
```
Enter your choice: 3
Enter doctor ID: 2
Appointment ID: 4
Patient ID: 7
Doctor ID: 2
Appointment Date: 2025-07-04 11:00:00
Description: Neurological evaluation
```

Schedule Appointment:



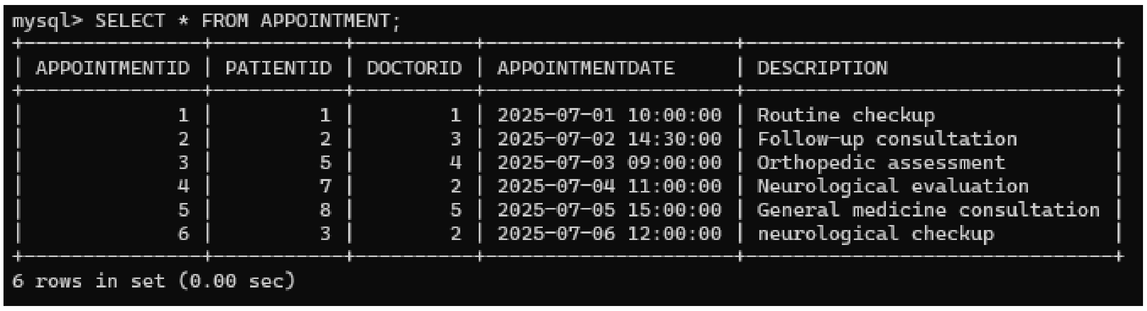
```
Enter your choice: 4
Enter patient ID: 3
Enter doctor ID: 2
Enter appointment date (YYYY-MM-DD HH:MM:SS): 2025-07-05 12:00:00
Enter description: neurological checkup
Appointment scheduled successfully.
```

Update Appointment:



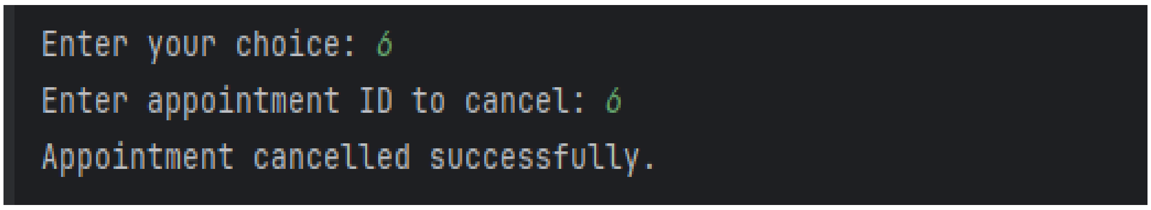
```
Enter your choice: 5
Enter appointment ID to update: 6
Enter patient ID: 3
Enter doctor ID: 2
Enter new appointment date (YYYY-MM-DD HH:MM:SS): 2025-07-06 12:00:00
Enter new description: neurological checkup
Appointment updated successfully.
```

Changed the date of appointment from 5/7/25 to 6/7/25

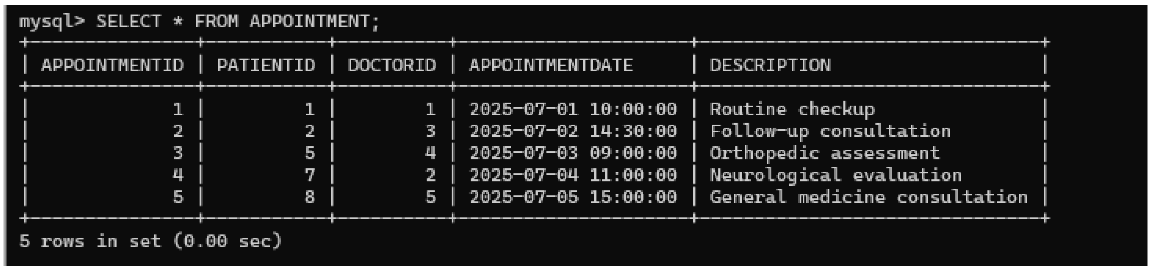


```
mysql> SELECT * FROM APPOINTMENT;
+---------------+-----------+----------+---------------------+-------------------------------+
| APPOINTMENTID | PATIENTID | DOCTORID | APPOINTMENTDATE     | DESCRIPTION                   |
+---------------+-----------+----------+---------------------+-------------------------------+
|             1 |         1 |        1 | 2025-07-01 10:00:00 | Routine checkup               |
|             2 |         2 |        3 | 2025-07-02 14:30:00 | Follow-up consultation        |
|             3 |         5 |        4 | 2025-07-03 09:00:00 | Orthopedic assessment         |
|             4 |         7 |        2 | 2025-07-04 11:00:00 | Neurological evaluation       |
|             5 |         8 |        5 | 2025-07-05 15:00:00 | General medicine consultation |
|             6 |         3 |        2 | 2025-07-06 12:00:00 | neurological checkup          |
+---------------+-----------+----------+---------------------+-------------------------------+
6 rows in set (0.00 sec)
```

Cancel Appointment:



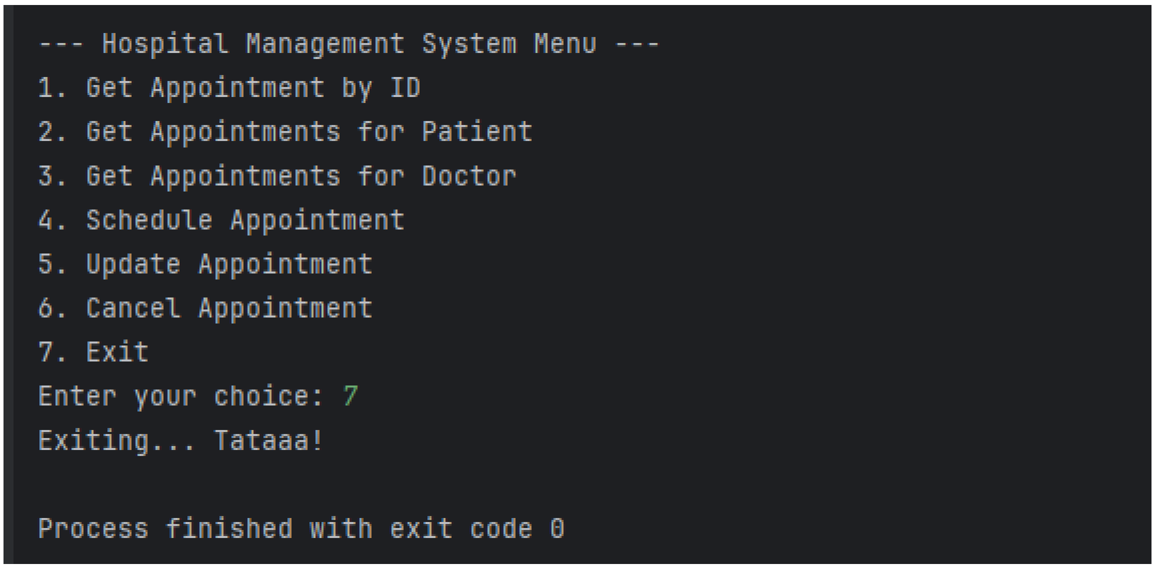
```
Enter your choice: 6
Enter appointment ID to cancel: 6
Appointment cancelled successfully.
```



```
mysql> SELECT * FROM APPOINTMENT;
+---------------+-----------+----------+---------------------+-------------------------------+
| APPOINTMENTID | PATIENTID | DOCTORID | APPOINTMENTDATE     | DESCRIPTION                   |
+---------------+-----------+----------+---------------------+-------------------------------+
|             1 |         1 |        1 | 2025-07-01 10:00:00 | Routine checkup               |
|             2 |         2 |        3 | 2025-07-02 14:30:00 | Follow-up consultation        |
|             3 |         5 |        4 | 2025-07-03 09:00:00 | Orthopedic assessment         |
|             4 |         7 |        2 | 2025-07-04 11:00:00 | Neurological evaluation       |
|             5 |         8 |        5 | 2025-07-05 15:00:00 | General medicine consultation |
+---------------+-----------+----------+---------------------+-------------------------------+
5 rows in set (0.00 sec)
```

Cancelled the created appointment

Exit:



```
--- Hospital Management System Menu ---
1. Get Appointment by ID
2. Get Appointments for Patient
3. Get Appointments for Doctor
4. Schedule Appointment
5. Update Appointment
6. Cancel Appointment
7. Exit
Enter your choice: 7
Exiting... Tataaa!

Process finished with exit code 0
```